

MULTIMEDIA



UNIVERSITY

STUDENT ID NO

--	--	--	--	--	--	--	--	--	--

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2019/2020

ECP2216 – MICROCONTROLLER AND MICROPROCESSOR SYSTEMS (TE / RE)

23 OCTOBER 2019
9.00 a.m – 11.00 a.m
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 13 pages including cover page with 5 Questions only.
2. Attempt **ALL FIVE COMPULSORY** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided.

Question 1

- (a) (i) von Neumann architecture is the basis of modern computing with stored-program capability which allows a general computer to run different programs with the same hardware. Using a suitable diagram, describe the concept of von Neumann architecture.
[6 marks]
- (ii) Compare microprocessor with microcontroller in terms of its hardware architecture, instruction set, application, and cost.
[4 marks]
- (b) (i) 80386DX Intel 32-bit microprocessor has six functional units. Name any **THREE** of these functional units and briefly explain their purposes.
[6 marks]
- (ii) List out **TWO** shortcomings of using four coupled single-core processors systems as compared to Quad-core processor systems.
[4 marks]

Continued

Question 2

- (a) The pin configuration of an 8051-microcontroller is as shown in Q2a.

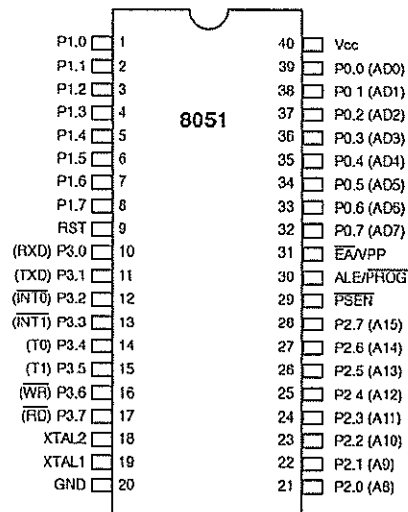


Figure Q2a: Pin configuration of 8051 microcontroller

Voldemort is an embedded system design engineer. He is currently handling a project to design an 8051-microcontroller system which requires the following external hardware.

- 32 kB of external data memory (RAM)
- 16 kB of external code memory (ROM)
- 3 push buttons
- 2 motor control signals
- Serial communication interface to computer

Advise Voldemort on how each pin in the 8051-microcontroller should be used. You may describe your advice in words.

[6 marks]

- (b) You are given two types of memory ICs. They are 16 Kbytes ROM and 16 Kbytes RAM. Expand the memory system (external) of an 8051-microcontroller for 16 Kbytes of program/code space and 16 Kbytes of data space. The memory system must be contiguous with the first 16 Kbytes the external space is for the program/code.

[8 marks]

Continued

- (c) Evaluate the content of Accumulator (ACC), B register, Register R0, Stack Pointer (SP), and Program Status Word (PSW), after the execution of each following instructions. Given the initial value of ACC = 88H, B = 99H, R0 = 34H, SP = 2FH, and PSW = 00H

```

MOV    R0, #82H
MOV    A, #22H
ADD    A, R0
PUSH   ACC
MOV    B, #00000100B
MUL    AB

```

Instruction	ACC	B	R0	SP	PSW
Initial	88H	99H	34H	2FH	00H
MOV R0, #82H					
MOV A, #22H					
ADD A, R0					
PUSH ACC					
MOV B, #00000100B					
MUL AB					

* Write down your answer in the answer script.

[6 marks]

Continued

Question 3

- (a) Determine the addressing modes for the source and the destination operand of the following MCS-51 instructions.

- i) SUBB A, @R0
- ii) MOV R7, #0FFH
- iii) JB 20H, \$
- iv) MOVC A, @A+PC

[4 marks]

- (b) Given a sequence of MCS-51 assembly language program below, complete the table below by finding the machine codes and its address, the total code size, and the total execution time in machine cycle.

```

ORG 0F00H
MOV A, P1
ANL A, #0FH
XRL A, #0FH
NOP
MOV P0, A
END

```

Instruction	Code Address	Machine code	Code size	Execution time (m. c.)
ORG 0F00H				
MOVA, P1				
ANL A, #0FH				
XRL A, #0FH				
NOP				
MOV P0, A				
TOTAL				

* Write down your answer in the answer script.

[6 marks]

Continued

- (c) In an 8051-microcontroller hardware system, in which,
- The P0 of an 8051-microcontroller is connected to an array of high-power LED driver. The high-power LED will light up by setting its respective I/O to logic '1'.
 - The P1.0 is connected to a toggle switch with the ON state sets the I/O pin to logic '1'.

Assuming the microcontroller is running with 12 MHz clock speed, using MCS-51 instruction sets,

- i) Write a delay subroutine named "DELAY_2MS" that would generate approximately 2 milliseconds of delay.
[3 marks]
- ii) Write another subroutine named "DELAY_500MS" that would generate approximately 500 milliseconds of delay. The subroutine should make use of "DELAY_2MS" subroutine in part (c)(i).
[3 marks]
- iii) Write a complete program that will blink the high-power LED array at an interval of 500 milliseconds **ONLY** when the toggle switch is turned ON. The main program shall include proper initialization of the I/O and any registers, if needed.
[4 marks]

Continued

Question 4

- (a) Consider an 8051-microcontroller system is to be designed to use two normally high push buttons labelled as PB0 and PB1. Whenever PB0 goes high-to-low, a square wave with period of 0.8ms will be generated from Port 1 pin 1 using Timer 1 interrupt. Whenever PB1 goes high-to-low, the square wave will be stopped and Port 1 pin 1 should be at low state. Both PB0 and PB1 are connected to external hardware interrupt INT0 pin and INT1 pin respectively. Assume that crystal frequency operates at 12MHz.
- (i) Determine the values of IE register, TMOD register, and reload value of Timer 1 (TH1 and TL1 registers) in hexadecimal format.
[4 marks]
- (ii) Write an MCS-51 assembly language program with *interrupts* to perform the abovementioned task.
[8 marks]
- (b) Consider an 8051-microcontroller system is to be configured as UART to operate at 9600 baud rate generated by Timer 1 and use 8-bit ASCII code, along with one stop bit. (Assume 11.0592MHz crystal frequency, SMOD = 0).
- (i) Determine the values of SCON register, TMOD register and reload value of Timer 1 (TH1 register) in hexadecimal format.
[3 marks]
- (ii) Write a MCS-51 assembly language program to initialize the registers in part (b)(i) and transmit ASCII character "A" continuously through the serial port.
[5 marks]

Continued

Question 5

Figure Q5 shows an 8051 microcontroller interfaces to two press buttons PB1 and PB2 on P0.1 and P0.2 respectively, a seven-segment LED display device on Port 1.0 to Port 1.7 and a *Mixer Machine* on Port 2.1 to Port 2.7.

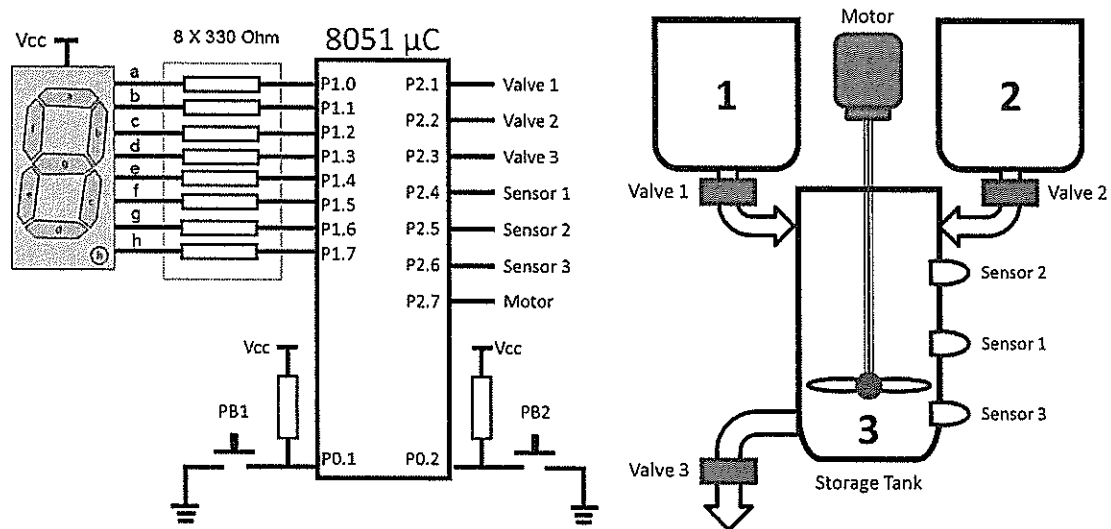


Figure Q5

Output lines of 8051 Microcontroller:

P2.1, P2.2 and P2.3 provide signals to the solenoid valves. A logical **HIGH** from the lines will open the corresponding valve.

P2.7 provide signal to the mixing motor which is activated by a logical **HIGH**.

Input lines of 8051 Microcontroller:

Three level sensors send signals into P2.4 to P2.6. A logical **HIGH** from the sensor indicates that it has detected the content in the container.

Two press buttons send signals into P0.1 to P0.2. A logical **LOW** from the press button indicates that it is pressed.

Table Q5:

It shows the bit patterns for each character of the word "*done*" to decode the seven-segment LED display.

Continued

Table Q5

Digit	<i>h</i>	<i>g</i>	<i>f</i>	<i>e</i>	<i>d</i>	<i>c</i>	<i>b</i>	<i>a</i>
	1	0	1	0	0	0	0	1
	1	0	1	0	0	0	1	1
	1	0	1	0	1	0	1	1
	1	0	0	0	0	1	1	0

- (a) Based on Figure Q5 and Table Q5, write a MCS-51 assembly language subroutine called **DONE** to perform the following tasks:

1. *Seven-segment LED display will repeatedly display the characters in sequence starting from d, o, n, and E.*
2. *Time duration for each character to be displayed is 0.5 second.*
3. *Assume 0.5 second delay subroutine called **POINTFIVESEC** is available.*
4. *The display will be turned off only if the button press on **PB2** occurs.*

[5 marks]

- (b) Based on Figure Q5, write a MCS-51 assembly language program to carry out the process:

1. *Initially, all valves, seven-segment LED display and motor are turned off.*
2. *When press button **PB1** is pressed and released, the storage tank is first filled with ingredient 1 through a solenoid **Valve 1**.*
3. *When the ingredient 1 reaches Level 1 (indicated by **Sensor 1**), **Valve 1** is closed.*
4. *Next, the storage tank is filled with ingredient 2 via **Valve 2** until mixture in the storage tank reaches Level 2 (indicated by **Sensor 2**), **Valve 2** is closed.*
5. *Then, the mixing **Motor** starts the stirring process that last for approximately 1 minute.*
6. *Finally, the dispenser **Valve 3** opens to dispense the mixture until Level 3 (indicated by **Sensor 3**) is reached.*
7. *Subroutine **DONE** in part (a) is called.*
8. *The whole process is repeated again from step 2 onwards.*

[15 marks]

End of Page

APPENDIX

Special Function Register Formats

Interrupt Enable (IE)

Bit Addr.	AFH	-	-	ACH	ABH	AAH	A9H	A8H
Name	EA	-	-	ES	ET1	EX1	ET0	EX0

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IE.7	EA	Global enable/disable. EA = 1, each individual source is enabled/disabled by setting/clearing its enable bit. EA = 0, disable all interrupts.
IE.6	-	Undefined
IE.5	-	Not implemented in 8051. ET2 for 8052.
IE.4	ES	Serial port interrupt enable bit.
IE.3	ET1	Timer 1 interrupt enable bit.
IE.2	EX1	External interrupt enable bit.
IE.1	ET0	Timer 0 interrupt enable bit.
IE.0	EX0	External interrupt enable bit.

Interrupt Priority (IP)

Bit Addr.	-	-	-	BCH	BBH	BAH	B9H	B8H
Name	-	-	-	PS	PT1	PX1	PT0	PX0

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IP.7	-	Undefined
IP.6	-	Undefined
IP.5	-	Not implemented in 8051. PT2 for 8052.
IP.4	PS	Serial port interrupt priority bit.
IP.3	PT1	Timer 1 interrupt priority bit.
IP.2	PX1	External interrupt priority bit.
IP.1	PT0	Timer 0 interrupt priority bit.
IP.0	PX0	External interrupt priority bit.

Interrupt Vectors

Interrupt Source	Flag	Vector Address
System Reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial Port	RI & TI	0023H
Timer 2 (8052)	TF2 or EXF2	002BH

Program Status Word (PSW)

Bit Addr.	D7H	D6H	D5H	D4H	D3H	D2H	-	D0H
Name	CY	AC	F0	RS1	RS0	OV	-	P

Serial Control (SCON)

Bit Addr.	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
Name	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

BIT	SYMBOL	FUNCTION
SCON.7	SM0	Serial port mode bit 0 (see Table A.1).
SCON.6	SM1	Serial port mode bit 1 (see Table A.1).
SCON.5	SM2	Serial port mode bit 2; enables multiprocessor communications in modes 2 and 3; RI will not be activated if received 9 th bit is 0. In mode 1, if SM2 = 1, then RI will be activated only if a valid stop bit was received. In mode 0, SM2 should be 0.
SCON.4	REN	Receiver enable; must be set to receive characters.
SCON.3	TB8	Transmit bit 8; 9 th bit transmitted in modes 2 and 3; set/cleared by software.
SCON.2	RB8	Receive bit 8; 9 th bit received.
SCON.1	TI	Transmit interrupt flag; set at end of character transmission; cleared by software.
SCON.0	RI	Receive interrupt flag; set at end of character reception; cleared by software.

Table A.1 The 8051 Serial Port Mode Selection

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift register	Fixed
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fixed
1	1	3	9-bit UART	Variable

Timer Control (TCON)

Bit Addr.	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

BIT	SYMBOL	FUNCTION
TCON.7	TF1	Timer-1 overflow flag. Set by hardware on overflow. Cleared by hardware when processor vectors to interrupt routine. Must be cleared by software when not involve interrupt
TCON.6	TR1	Timer-1 run control bit. Set/cleared by software to turn timer/counter on/off.
TCON.5	TF0	Timer-0 overflow flag. Do the same function as TF1 but for Timer-0.
TCON.4	TR0	Timer-0 run control bit. Do the same function as TR1 but for Timer-0.
TCON.3	IE1	External interrupt-1 edge flag. Set by hardware when interrupt-1 falling edge is detected. Cleared by hardware when interrupt is processed.
TCON.2	IT1	Interrupt-1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TCON.1	IE0	External interrupt-0 edge flag. Do the same function as IE1 but for external interrupt-0.
TCON.0	IT0	Interrupt-0 Type control bit. Do the same function as IT1 but for external interrupt-0.

Timer Mode (TMOD)

Bit	7	6	5	4	3	2	1	0
Name	GATE	C/T	M1	M0	GATE	C/T	M1	M0

BIT	SYMBOL	FUNCTION
TMOD.7	GATE1	When this bit is set the timer will only run when INT1 (P3.3) is high (hardware control). When this bit is cleared the timer will run regardless of the state of INT1 (software control).
TMOD.6	C/T1	Timer / Counter select bit. $C / \bar{T} = 0 \rightarrow$ Timer operation. $C / \bar{T} = 1 \rightarrow$ Counter operation.
TMOD.5	M1	Mode selection bits (see Table A.2). [for timer 1]
TMOD.4	M0	Mode selection bits (see Table A.2). [for timer 1]
TMOD.3	GATE0	Exactly the same function as GATE1 but for Timer0
TMOD.2	C/T0	Exactly the same function as C/T1 but for Timer0
TMOD.1	M1	Mode selection bits (see Table A.2). [for timer 0]
TMOD.0	M0	Mode selection bits (see Table A.2). [for timer 0]

Table A.2 Timer Mode Selection

M1	M0	Timer Mode	Description of Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer
1	0	2	8-bit auto-reload
1	1	3	Split timer mode

MCS-51 Opcode Map

Byte Instruction operands	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP 1B	JBC bit, rel 2B	JB bit, rel 2B	JNB bit, rel 2B	JC rel 2B	JNC rel 2B	JZ rel 2B	JNZ rel 2B	SJMP rel 2B	MOV DPTR, #data16 2B	ORL C, /bit 2B	ANL C, /bit 2B	PUSH dir 2B	POP dir 2B	MOVX A, @DPTR 2B	MOVX @DPTR, A 2B
1	AJMP (P0) 2B	ACALL (P0) 2B	AJMP (P1) 2B	ACALL (P1) 2B	AJMP (P2) 2B	ACALL (P2) 2B	AJMP (P3) 2B	ACALL (P3) 2B	AJMP (P4) 2B	ACALL (P4) 2B	AJMP (P5) 2B	ACALL (P5) 2B	AJMP (P6) 2B	ACALL (P6) 2B	AJMP (P7) 2B	ACALL (P7) 2B
2	LJMP addr16 2B	LCALL addr16 2B	RET 1B	RET 1B	ORL dir, A 1B	ORL dir, A 1B	XRL dir, A 1B	XRL dir, A 1B	ANL C, bit 2B	MOV bit, C 2B	MOV C, bit 1B	CPL bit 1B	CLR bit 1B	SETB bit 1B	MOVX A, @R0 2B	MOVX @R0, A 2B
3	RR A 1B	RRC A 1B	RL A 1B	RLC A 1B	ORL A, #data 1B	ORL A, #data 1B	XRL A, #data 1B	XRL A, #data 1B	MOV A, #data 1B	MOV A, #data 1B	INC DPTR 1B	CPL C 1B	CLR C 1B	SETB C 1B	MOVX @R1, A 2B	MOVX @R1, A 2B
4	INC A 1B	DEC A 1B	ADD A, #data 1B	ADDC A, #data 1B	ORL A, #data 1B	ORL A, #data 1B	XRL A, #data 1B	XRL A, #data 1B	MOV A, #data 1B	MOV A, #data 1B	MUL AB 1B	CJNE A, #data, rel 2B	SWAP A 1B	DA A 1B	CLR A 1B	CPL A 1B
5	INC dir 1B	DEC dir 1B	ADD A, dir 1B	ADDC A, dir 1B	ORL A, dir 1B	ORL A, dir 1B	XRL A, dir 1B	XRL A, dir 1B	MOV dir, A 1B	MOV dir, A 1B		CJNE A, dir, rel 2B	XCH A, dir 1B	DJNZ dir, rel 1B	MOV dir, A 1B	MOV dir, A 1B
6	INC @R0 1B	DEC @R0 1B	ADD A, @R0 1B	ADDC A, @R0 1B	ORL A, @R0 1B	ORL A, @R0 1B	XRL A, @R0 1B	XRL A, @R0 1B	MOV @R0, dir 1B	MOV @R0, dir 1B	MOV @R0, dir 1B	CJNE @R0, #data, rel 2B	XCH A, @R0 1B	XCHD A, @R0 1B	MOV @R0, A 1B	MOV @R0, A 1B
7	INC @R1 1B	DEC @R1 1B	ADD A, @R1 1B	ADDC A, @R1 1B	ORL A, @R1 1B	ORL A, @R1 1B	XRL A, @R1 1B	XRL A, @R1 1B	MOV @R1, dir 1B	MOV @R1, dir 1B	MOV @R1, dir 1B	CJNE @R1, #data, rel 2B	XCH A, @R1 1B	XCHD A, @R1 1B	MOV @R1, A 1B	MOV @R1, A 1B
8	INC R0 1B	DEC R0 1B	ADD A, R0 1B	ADDC A, R0 1B	ORL A, R0 1B	ORL A, R0 1B	XRL A, R0 1B	XRL A, R0 1B	MOV R0, dir 1B	MOV R0, dir 1B	MOV R0, dir 1B	CJNE R0, #data, rel 2B	XCH A, R0 1B	DJNZ R0, rel 1B	MOV R0, A 1B	MOV R0, A 1B
9	INC R1 1B	DEC R1 1B	ADD A, R1 1B	ADDC A, R1 1B	ORL A, R1 1B	ORL A, R1 1B	XRL A, R1 1B	XRL A, R1 1B	MOV R1, dir 1B	MOV R1, dir 1B	MOV R1, dir 1B	CJNE R1, #data, rel 2B	XCH A, R1 1B	DJNZ R1, rel 1B	MOV R1, A 1B	MOV R1, A 1B
A	INC R2 1B	DEC R2 1B	ADD A, R2 1B	ADDC A, R2 1B	ORL A, R2 1B	ORL A, R2 1B	XRL A, R2 1B	XRL A, R2 1B	MOV R2, dir 1B	MOV R2, dir 1B	MOV R2, dir 1B	CJNE R2, #data, rel 2B	XCH A, R2 1B	DJNZ R2, rel 1B	MOV R2, A 1B	MOV R2, A 1B
B	INC R3 1B	DEC R3 1B	ADD A, R3 1B	ADDC A, R3 1B	ORL A, R3 1B	ORL A, R3 1B	XRL A, R3 1B	XRL A, R3 1B	MOV R3, dir 1B	MOV R3, dir 1B	MOV R3, dir 1B	CJNE R3, #data, rel 2B	XCH A, R3 1B	DJNZ R3, rel 1B	MOV R3, A 1B	MOV R3, A 1B
C	INC R4 1B	DEC R4 1B	ADD A, R4 1B	ADDC A, R4 1B	ORL A, R4 1B	ORL A, R4 1B	XRL A, R4 1B	XRL A, R4 1B	MOV R4, dir 1B	MOV R4, dir 1B	MOV R4, dir 1B	CJNE R4, #data, rel 2B	XCH A, R4 1B	DJNZ R4, rel 1B	MOV R4, A 1B	MOV R4, A 1B
D	INC R5 1B	DEC R5 1B	ADD A, R5 1B	ADDC A, R5 1B	ORL A, R5 1B	ORL A, R5 1B	XRL A, R5 1B	XRL A, R5 1B	MOV R5, dir 1B	MOV R5, dir 1B	MOV R5, dir 1B	CJNE R5, #data, rel 2B	XCH A, R5 1B	DJNZ R5, rel 1B	MOV R5, A 1B	MOV R5, A 1B
E	INC R6 1B	DEC R6 1B	ADD A, R6 1B	ADDC A, R6 1B	ORL A, R6 1B	ORL A, R6 1B	XRL A, R6 1B	XRL A, R6 1B	MOV R6, dir 1B	MOV R6, dir 1B	MOV R6, dir 1B	CJNE R6, #data, rel 2B	XCH A, R6 1B	DJNZ R6, rel 1B	MOV R6, A 1B	MOV R6, A 1B
F	INC R7 1B	DEC R7 1B	ADD A, R7 1B	ADDC A, R7 1B	ORL A, R7 1B	ORL A, R7 1B	XRL A, R7 1B	XRL A, R7 1B	MOV R7, dir 1B	MOV R7, dir 1B	MOV R7, dir 1B	CJNE R7, #data, rel 2B	XCH A, R7 1B	DJNZ R7, rel 1B	MOV R7, A 1B	MOV R7, A 1B